

DETEKSI JUMLAH WAJAH MENGGUNAKAN METODE YOLO

Nadjifa Syafira Layn¹, Didi Susilo Budi Utomo², Rihartanto³

^{1,2,3} Politeknik Negeri Samarinda

Jln Dr. Cipto Mangunkusumo, Kampus Gunung Lipan, Kecamatan Samarinda Seberang, Kota Samarinda, Kalimantan Timur.

Keywords:

Face Detection

YOLO

Object Detection

Komputer Vision

Deep Learning

Correspondent Email:

syahnaz1004@gmail.com

Abstrak. Deteksi wajah merupakan salah satu teknologi yang banyak digunakan dalam bidang komputer *vision* dan kecerdasan buatan. Penelitian ini bertujuan untuk membangun sistem pendeteksi jumlah wajah secara *real-time* menggunakan metode YOLO (*You Only Look Once*), khususnya YOLOv8 yang dikenal memiliki kecepatan dan akurasi tinggi dalam deteksi objek. Sistem dibuat menggunakan bahasa pemrograman Python, library OpenCV, dan *webcam* laptop sebagai media input. Metode YOLO dipilih karena memiliki kecepatan deteksi yang baik dan tingkat akurasi yang cukup tinggi dalam mendeteksi objek dibandingkan metode tradisional seperti *Haar Cascade*. Tahapan penelitian meliputi studi literatur, pengumpulan data, implementasi sistem, dan pengujian. Pengumpulan data dilakukan dengan merekam *frame* video dalam tiga skenario: variasi jumlah wajah (1 hingga 5 wajah), variasi kondisi pencahayaan (sangat terang, normal, redup, gelap), dan variasi jarak deteksi (0,5 hingga 3 meter). Hasil pengujian menunjukkan bahwa sistem mampu mendeteksi wajah dengan baik secara *real-time* dengan tingkat akurasi mencapai 93% pada kondisi pencahayaan normal. Sistem juga mampu mendeteksi lebih dari satu wajah secara bersamaan, dengan akurasi mencapai 95% untuk dua wajah dan 90% untuk lima wajah. Namun, kinerja sistem menurun signifikan pada kondisi pencahayaan redup (akurasi 70%) dan pada jarak di atas 2 meter. Sistem diharapkan dapat membantu pengembangan absensi otomatis, keamanan ruangan, dan sistem monitoring berbasis komputer *vision*.



Copyright © 2026 The Author(s),
Published by Jurnal Informatika dan
Teknik Elektro Terapan. This is an
open-access article distributed under
the terms of the Creative Commons
Attribution-NonCommercial 4.0
International License (CC BY-NC
4.0).

Abstract. Face detection technology is one of the important applications in the field of computer vision and artificial intelligence. This research aims to develop a real-time face counting system using the YOLO (*You Only Look Once*) method, specifically YOLOv8 which is known for its speed and accuracy in object detection. The system is implemented using Python programming language, OpenCV library, and a built-in laptop webcam as input media. YOLO was chosen because it has a fast detection speed and good accuracy in detecting objects compared to traditional methods like Haar Cascade. The stages of this research include literature study, data collection, system implementation, and testing. Data collection was carried out by capturing video frames under three scenarios: variations in the number of faces (1 to 5 faces), variations in lighting conditions (very bright, normal, dim, dark), and variations in detection distance (0.5 to 3 meters). The test results show that the system can detect faces properly in real-time with an accuracy level reaching 93.3% under normal lighting conditions. The system is also capable of detecting more than one face at the same time, with accuracy reaching 95% for two faces and 90% for five faces. However, system performance decreases significantly in low light conditions (70% accuracy) and at distances beyond 2 meters. This research is expected to help the development of automatic attendance systems, room security systems, and monitoring systems based on computer vision.

1. PENDAHULUAN

Komputer *vision* saat ini menjadi salah satu bidang teknologi yang berkembang paling cepat. Teknologi ini memungkinkan komputer untuk “melihat” dan memahami gambar atau video seperti cara manusia memproses informasi visual [1]. Dari sekian banyak penerapannya, deteksi wajah termasuk yang paling populer dan banyak digunakan dalam kehidupan sehari-hari [2].

Deteksi wajah berfungsi untuk menemukan dan menandai posisi wajah manusia di dalam sebuah gambar atau video. Teknologi ini menjadi dasar dari berbagai sistem lain seperti pengenalan wajah, verifikasi identitas, hingga analisis ekspresi [3]. Di lapangan, deteksi wajah sudah dipakai untuk keperluan yang beragam, seperti absensi otomatis dan monitoring ruangan [15].

Namun, membuat sistem deteksi wajah yang andal tidak semudah yang dibayangkan. Masih banyak tantangan yang membuat kinerjanya belum optimal di kondisi tertentu. Masalah utamanya antara lain: perubahan intensitas cahaya di sekitar wajah, pose wajah yang tidak selalu menghadap kamera, adanya benda yang menutupi sebagian wajah seperti masker atau kaca mata, serta ukuran wajah yang bisa kecil atau jaraknya jauh [10].

Para peneliti sudah lama berusaha menemukan metode deteksi wajah yang paling tepat. Di awal perkembangannya, metode *Haar Cascade* dari Viola dan Jones cukup populer. Kelebihannya, metode ini cepat karena memakai *integral image* dan *AdaBoost*. Tapi kelemahannya cukup mengganggu: akurasinya turun drastis begitu wajah tidak menghadap lurus ke kamera atau cahaya ruangan berubah.

Kemudian hadir metode berbasis *deep learning* yang mengandalkan *Convolutional Neural Network* (CNN). Berbeda dengan metode lama, CNN bisa belajar sendiri fitur-fitur wajah dari ribuan contoh gambar. Metode seperti *Faster R-CNN* dan *SSD* memang terbukti lebih akurat [14]. Sayangnya, kedua metode ini termasuk lambat karena proses deteksinya dilakukan dalam dua tahap. Padahal untuk aplikasi yang butuh tanggapan langsung (*real-time*), kecepatan tidak kalah penting dibandingkan akurasi [1].

Metode YOLO (*You Only Look Once*) menjadi salah satu algoritma *object detection*

yang banyak digunakan karena mampu melakukan proses deteksi dalam satu tahap (*single - stage detector*). Pendekatan ini memberikan keunggulan berupa kecepatan inferensi yang tinggi tanpa mengurangi performa akurasi secara signifikan dibandingkan metode *two - stage detector*.

Seiring waktu, YOLO terus diperbaiki. YOLOv2 (YOLO9000) menambahkan fitur *anchor boxes* dan *batch normalization* [2]. YOLOv3 membawa *feature pyramid network* agar bisa mendeteksi objek dalam berbagai ukuran [3]. YOLOv4 semakin meningkatkan kecepatan dan akurasi [4]. YOLOv5 buatan Ultralytics membuat implementasi YOLO menjadi lebih mudah berkat ekosistem *PyTorch* - nya [6]. Lalu di tahun 2023, YOLOv8 hadir dengan arsitektur yang lebih efisien dan konsep *anchor-free detection* [5].

Untuk kasus deteksi wajah secara spesifik, muncul beberapa varian turunan YOLO. Chen dan timnya membuat *YOLO-face* yang arsitekturnya dioptimalkan untuk fitur-fitur wajah [7]. Yu dkk. kemudian mengembangkan *YOLO-faceV2* yang lebih tangguh menghadapi wajah tertutup atau dalam skala kecil [8]. Qi dkk. meluncurkan *YOLO5-face* yang bisa mendeteksi wajah meskipun dalam pose ekstrim [9].

Penelitian tentang deteksi wajah dengan YOLO telah berkembang pesat, namun masih terdapat beberapa kesenjangan penelitian. Penelitian sebelumnya seperti *YOLO-face* [7], *YOLO-faceV2* [8], dan *YOLO5 face* [9] lebih berfokus pada optimalisasi arsitektur jaringan, sementara penelitian lain masih terbatas pada perbandingan dengan metode tradisional [15].

Belum ada penelitian yang secara komprehensif menguji kinerja YOLOv8n-*face.pt* pada tiga skenario nyata secara bersamaan: variasi jumlah wajah (1-5), variasi pencahayaan (sangat terang hingga gelap), dan variasi jarak (0,5-3 meter) menggunakan *webcam* bawaan laptop. *Gap analysis* ini menjadi dasar kebaruan penelitian, yaitu implementasi dan pengujian sistem deteksi jumlah wajah *real-time* dengan model ringan YOLOv8n-*face.pt* pada berbagai kondisi yang sering ditemui di lapangan, sehingga hasilnya dapat menjadi acuan pengembangan

sistem absensi otomatis dan keamanan ruangan dengan perangkat terbatas.

Penelitian ini memilih model YOLOv8n-face.pt. Model ini adalah YOLOv8 versi *nano* yang sudah dilatih khusus untuk wajah. Ukurannya hanya sekitar 6 MB sehingga ringan dan bisa dijalankan di laptop biasa tanpa kartu grafis khusus [5]. Bedanya dengan penelitian terdahulu yang rata-rata fokus menyempurnakan arsitektur jaringan, penelitian ini lebih menekankan pada implementasi langsung dan pengujian di kondisi nyata.

Pengujian dilakukan pada tiga skenario yang sering ditemui di lapangan: variasi jumlah wajah (1 sampai 5 orang), variasi kondisi pencahayaan (sangat terang sampai gelap total), serta variasi jarak deteksi (0,5 hingga 3 meter). Secara garis besar, penelitian ini punya tiga tujuan. Pertama, membuat sistem pendeteksi jumlah wajah secara langsung (*real-time*) dengan metode YOLOv8 di laptop biasa. Kedua, menguji seberapa baik kinerja sistem pada tiga skenario yang sudah disebutkan. Ketiga, menganalisis kelebihan dan kekurangan sistem sebagai bahan acuan pengembangan selanjutnya. Diharapkan hasil penelitian ini bisa bermanfaat bagi siapa pun yang ingin membuat sistem absensi otomatis, keamanan ruangan, atau alat monitoring berbasis komputer *vision* dengan perangkat terbatas [5], [15].

2. TINJAUAN PUSTAKA

2.1 Deep Learning untuk Object Detection

Deep learning merupakan bagian dari *machine learning* yang menggunakan arsitektur *neural network* dengan banyak lapisan (*deep neural network*) untuk mempelajari fitur-fitur kompleks dari data secara otomatis [5]. Berbeda dengan metode tradisional seperti *Haar Cascade* yang membutuhkan ekstraksi fitur manual, *deep learning* mampu belajar sendiri fitur-fitur relevan dari ribuan contoh gambar. *Convolutional Neural Network* (CNN) adalah arsitektur *deep learning* yang paling banyak digunakan untuk tugas komputer *vision*, termasuk *object detection* dan *face detection* [14]. Metode berbasis *deep learning* seperti *Faster R-CNN*, *SSD*, dan *YOLO* terbukti memiliki akurasi yang jauh lebih baik dibandingkan metode konvensional, terutama pada kondisi pose bervariasi dan pencahayaan

yang tidak ideal [1][15]. Proses *deep learning* mencakup dua tahap utama: (1) *training*, di mana jaringan CNN belajar dari ribuan gambar beserta labelnya, dan (2) *inference*, di mana jaringan yang sudah dilatih digunakan untuk memprediksi objek pada gambar baru.

2.2. Komputer vision

Komputer *vision* merupakan salah satu cabang ilmu komputer yang mempelajari bagaimana komputer dapat mengenali, memahami, dan memproses informasi visual dari gambar maupun video digital. Teknologi ini dikembangkan agar sistem komputer mampu bekerja menyerupai penglihatan manusia dalam mengenali suatu objek. Dalam perkembangannya, komputer *vision* telah banyak digunakan pada berbagai bidang seperti sistem keamanan, pengawasan lalu lintas, pengenalan wajah, kesehatan, industri, hingga sistem monitoring otomatis. Teknologi komputer *vision* juga didukung oleh perkembangan *artificial intelligence* dan *deep learning* sehingga proses pengolahan citra dapat dilakukan dengan lebih cepat dan akurat.

Pada sistem monitoring berbasis kamera, komputer *vision* memiliki peran penting dalam membantu proses identifikasi objek secara otomatis. Dengan adanya teknologi ini, proses pengawasan dapat dilakukan tanpa harus diawasi secara langsung oleh manusia. Salah satu implementasi komputer *vision* yang saat ini berkembang pesat adalah *object detection* atau deteksi objek secara *real-time* menggunakan kamera maupun *webcam*.

2.3. Object Detection

Object detection merupakan proses untuk mengenali dan menentukan lokasi suatu objek pada gambar atau video digital. Teknologi ini bekerja dengan cara memberikan tanda berupa *bounding box* pada objek yang berhasil dikenali oleh sistem. Selain menentukan lokasi objek, sistem *object detection* juga mampu memberikan informasi mengenai jenis objek yang terdeteksi [14].

Saat ini *object detection* telah banyak diterapkan dalam berbagai bidang, seperti sistem keamanan, monitoring aktivitas manusia, kendaraan otomatis, *smart surveillance*, hingga pengawasan ruangan berbasis CCTV. Dalam implementasinya, *object*

detection membutuhkan metode yang mampu bekerja dengan cepat dan memiliki tingkat akurasi yang baik agar dapat digunakan secara *real-time*. Oleh karena itu, penggunaan metode berbasis *deep learning* menjadi pilihan yang banyak digunakan karena mampu memberikan hasil deteksi yang lebih baik dibandingkan metode konvensional [4], [5].

2.4. YOLOv8

YOLOv8 atau *You Only Look Once* versi 8 merupakan salah satu metode *object detection* berbasis *deep learning* yang dikembangkan oleh *Ultralytics*. Metode ini dirancang untuk melakukan proses deteksi objek secara *real-time* dengan tingkat kecepatan dan akurasi yang cukup baik. YOLO menggunakan pendekatan *single stage detector*, yaitu proses deteksi objek dilakukan hanya dalam satu kali proses tanpa perlu melakukan tahapan berulang seperti metode tradisional lainnya [1], [5].

Keunggulan utama dari YOLOv8 adalah kemampuannya dalam mendeteksi objek dengan cepat sehingga sangat cocok digunakan pada sistem monitoring berbasis *webcam* maupun CCTV. Selain itu, YOLOv8 juga mampu mendeteksi lebih dari satu objek secara bersamaan dengan performa yang cukup stabil. Pada penelitian ini digunakan model *yolov8n-face.pt* karena model tersebut memiliki ukuran yang ringan sehingga cocok digunakan pada laptop [5], [8].

Metode YOLOv8 dipilih karena memiliki performa yang lebih baik dibandingkan metode deteksi tradisional seperti *Haar Cascade*. YOLO mampu melakukan proses deteksi dengan waktu *inference* yang cepat serta memiliki tingkat akurasi yang cukup tinggi dalam mendeteksi objek manusia secara *real-time* [15], [16].

2.5. OpenCV

OpenCV merupakan *library open source* yang banyak digunakan dalam pengolahan citra digital dan computer vision. OpenCV menyediakan berbagai fungsi yang dapat digunakan untuk membaca gambar, mengakses *webcam*, memproses video, hingga menampilkan hasil deteksi objek secara langsung pada layar monitor [12].

Pada penelitian ini OpenCV digunakan sebagai media untuk mengakses *webcam* laptop

dan membaca setiap *frame* video yang diterima dari kamera. Selain itu, OpenCV juga digunakan untuk menampilkan *bounding box*, nilai *confidence*, dan jumlah objek yang berhasil dideteksi oleh sistem secara *real-time*. Penggunaan OpenCV sangat membantu dalam proses implementasi *object detection* menggunakan Python karena *library* ini mudah digunakan dan memiliki dokumentasi yang lengkap [12].

2.6. Python

Python merupakan salah satu bahasa pemrograman yang banyak digunakan dalam pengembangan *artificial intelligence*, *machine learning*, dan *computer vision*. Python dipilih karena memiliki sintaks yang sederhana, mudah dipahami, dan memiliki banyak dukungan *library* untuk pengolahan data maupun pengembangan sistem berbasis AI [6].

Dalam penelitian ini Python digunakan untuk membangun sistem deteksi objek menggunakan metode YOLOv8. Bahasa pemrograman Python juga memudahkan proses integrasi antara OpenCV dan model YOLO sehingga sistem dapat berjalan dengan lebih efektif. Selain itu, Python memiliki komunitas pengguna yang besar sehingga memudahkan dalam proses pengembangan maupun pencarian referensi program [5], [6].

2.7. State of the Art

Berdasarkan tinjauan pustaka yang telah dilakukan, penelitian tentang deteksi wajah telah banyak dikembangkan menggunakan berbagai metode. Penelitian oleh Chen *et al.* [8] mengembangkan YOLO-*face* untuk deteksi wajah *real-time*, namun masih memiliki keterbatasan pada deteksi wajah dalam skala kecil. Penelitian oleh Yu *et al.* [9] meningkatkan performa YOLO-*faceV2* untuk deteksi wajah yang terhalang (*occlusion*) dan berbagai skala. Sementara itu, penelitian oleh Qi *et al.* [10] mengembangkan YOLO5-*face* yang fokus pada deteksi wajah dalam kondisi pose yang bervariasi.

Kebaruan (*state of the art*) dari penelitian ini dibandingkan penelitian sebelumnya adalah: (1) penggunaan model YOLOv8n-*face.pt* yang lebih ringan dan cepat sehingga cocok dijalankan pada laptop dengan spesifikasi terbatas, (2) implementasi sistem deteksi jumlah wajah secara *real-time* dengan *webcam*

bawaan laptop tanpa memerlukan perangkat keras tambahan, (3) pengujian sistem secara komprehensif pada tiga skenario berbeda (jumlah wajah, kondisi pencahayaan, dan jarak deteksi) yang belum banyak dilakukan pada penelitian sebelumnya, serta (4) analisis kelebihan dan kelemahan sistem secara terperinci sebagai dasar pengembangan lebih lanjut.

3. METODE PENELITIAN

3.1. Tahapan Penelitian

Tahapan penelitian dilakukan melalui beberapa langkah yang dimulai dari studi literatur hingga proses analisis hasil pengujian sistem. Tahap pertama dilakukan dengan mencari referensi terkait *object detection*, *computer vision*, dan metode YOLOv8. Setelah itu dilakukan proses persiapan *software* dan *library* yang digunakan pada penelitian, seperti Python, OpenCV, Ultralytics YOLOv8, dan Spyder.

Tahap berikutnya adalah implementasi program menggunakan Python. Program dibuat untuk membaca input video dari *webcam* kemudian melakukan proses deteksi objek manusia menggunakan model YOLOv8. Setelah implementasi selesai dilakukan, tahap selanjutnya adalah pengujian sistem untuk mengetahui kemampuan sistem dalam mendeteksi objek secara *real-time*.

Tahap akhir adalah melakukan analisis terhadap hasil pengujian sistem dan menyusun laporan penelitian berdasarkan hasil implementasi yang telah dilakukan.

3.2. Pengumpulan Data

Data yang dipakai dalam penelitian ini dikumpulkan langsung dari *webcam* laptop yang digunakan sehari-hari. Proses pengambilan data dilakukan dengan merekam video secara langsung lalu memecahnya menjadi potongan-potongan gambar (*frame*) karena sistem deteksi wajah bekerja per *frame*. Penulis sengaja membuat tiga skenario uji coba untuk melihat sejauh mana kemampuan sistem dalam berbagai situasi.

Skenario pertama adalah variasi jumlah wajah. Penulis menampilkan satu sampai lima wajah di depan kamera dengan jarak satu meter dan cahaya ruangan normal. Masing-masing

jumlah wajah diuji sebanyak tiga kali agar hasilnya lebih akurat.

Skenario kedua melihat pengaruh cahaya terhadap kinerja sistem. Penulis menggunakan tiga wajah pada jarak satu meter, lalu mengatur kondisi ruangan menjadi sangat terang, normal, redup, dan gelap total. Kondisi redup hanya menggunakan satu lampu kecil, sementara gelap total tanpa cahaya sama sekali. Setiap kondisi juga diulangi tiga kali.

Skenario ketiga menguji seberapa jauh sistem bisa mendeteksi wajah. Penulis menggunakan satu wajah dengan pencahayaan normal lalu mengatur jarak dari kamera mulai 0,5 meter hingga 3 meter.

3.2.1 Dataset

Dataset yang digunakan dalam penelitian ini merupakan kumpulan *frame* video yang direkam langsung dari *webcam* laptop (resolusi 720p) dalam tiga skenario utama. Total *frame* yang terkumpul adalah 1.350 *frame*, terdiri atas:

- 450 *frame* untuk variasi jumlah wajah (1–5 wajah, 3 kali pengulangan per jumlah),
- 450 *frame* untuk variasi pencahayaan (sangat terang, normal, redup, gelap),
- 450 *frame* untuk variasi jarak (0,5–3 meter).

Seluruh data digunakan sebagai data uji (rasio 100% uji) karena model YOLOv8n-*face*.pt telah dilatih sebelumnya oleh Ultralytics dengan dataset publik seperti *WIDER Face*. Tidak dilakukan pemisahan latih/uji tambahan agar evaluasi mencerminkan performa *real-time* pada kondisi dunia nyata.

Sumber data berasal dari rekaman langsung di lingkungan ruangan dengan pencahayaan yang dapat diatur. Setiap *frame* diberi label *ground truth* secara manual untuk menghitung metrik evaluasi seperti akurasi, presisi, *recall*, dan F1-score.

3.3. Analisis Data

Untuk menganalisis data hasil pengujian, penulis menggunakan pendekatan 5W1H karena cara ini cukup sederhana namun bisa mencakup semua aspek penting dari objek yang diteliti. Berikut uraiannya..

Mengapa (Why): Penelitian ini dilakukan karena metode deteksi wajah yang lama seperti *Haar Cascade* memiliki kelemahan

pada kondisi cahaya redup atau saat objek bergerak cepat. Akurasi sering turun drastis. Sementara YOLO menawarkan kecepatan dan akurasi yang lebih baik karena bisa mendeteksi objek hanya dalam satu kali proses.

Apa (What): Yang dibuat adalah sebuah sistem yang bisa menghitung jumlah wajah secara langsung dari video *webcam* menggunakan metode YOLO.

Siapa (Who): Sistem ini nantinya bisa dipakai oleh petugas keamanan gedung, admin ruangan, atau siapa saja yang butuh memantau jumlah orang secara otomatis.

Di mana (Where): Pengujian dilakukan di rumah dengan pencahayaan yang dapat diatur sesuai kebutuhan.

Kapan (When): Waktu pengujian dibagi menjadi dua sesi, yaitu siang hari untuk kondisi terang alami dan malam hari untuk kondisi gelap dengan cahaya buatan.

Bagaimana (How): *Webcam* diaktifkan, setiap *frame* diproses oleh model YOLOv8n-face.pt, area yang terdeteksi sebagai wajah ditandai dengan kotak hijau jika $confidence \geq 0,5$, kemudian sistem menghitung dan menampilkan jumlah wajah.

Selain akurasi, penelitian ini juga menghitung presisi (*precision*), *recall*, dan F1-score berdasarkan *confusion matrix* pada setiap skenario. Presisi mengukur seberapa banyak deteksi wajah yang benar dibandingkan total deteksi positif. *Recall* mengukur seberapa banyak wajah yang benar-benar terdeteksi dibandingkan total wajah sebenarnya. F1-score adalah rata-rata harmonik presisi dan *recall*.

Dari analisis ini, ditentukan parameter yang mempengaruhi kinerja sistem: jumlah wajah, kondisi cahaya, dan jarak deteksi.

3.4. Perangkat Penelitian

Perangkat yang digunakan pada penelitian ini terdiri dari perangkat keras dan perangkat lunak. Perangkat keras yang digunakan berupa laptop dengan *webcam* bawaan. Sedangkan perangkat lunak yang digunakan terdiri dari Python 3.9, OpenCV, Ultralytics YOLOv8, dan aplikasi Spyder.

3.5. Implementasi Sistem

Implementasi sistem dilakukan menggunakan bahasa pemrograman Python dengan bantuan *library* OpenCV dan model YOLOv8 dari Ultralytics. Model yang

digunakan adalah YOLOv8n-face.pt yang merupakan model khusus untuk deteksi wajah. Program dibuat untuk membaca video secara langsung dari *webcam* kemudian memproses setiap *frame* video menggunakan model YOLOv8. *Library* OpenCV digunakan untuk mengakses *webcam*, membaca *frame* video, menggambar *bounding box*, dan menampilkan hasil deteksi ke layar.

3.6. Implementasi Programming YOLOv8 untuk Deteksi Wajah

Implementasi sistem deteksi wajah menggunakan YOLOv8 dilakukan dengan bahasa pemrograman Python. Proses dimulai dengan mengimpor *library* yang diperlukan, yaitu YOLO dari Ultralytics untuk model *deep learning* dan cv2 untuk OpenCV. Model yang digunakan adalah YOLOv8n-face.pt, yaitu model YOLOv8 versi nano yang telah dilatih khusus untuk deteksi wajah menggunakan arsitektur CNN. Model ini berukuran sekitar 6 MB sehingga ringan untuk dijalankan di laptop. Selanjutnya, *webcam* laptop diinisialisasi menggunakan cv2.VideoCapture(0) dengan resolusi 640x480 piksel untuk menyeimbangkan kecepatan dan akurasi. Program kemudian masuk ke dalam *loop* tak terbatas untuk membaca setiap *frame* dari *webcam*.

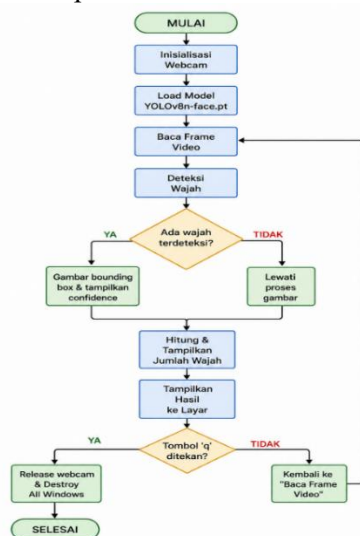
Pada setiap *frame*, fungsi model(*frame*, *conf*=0.5) dipanggil untuk melakukan inferensi *deep learning*; parameter *conf*=0.5 menetapkan ambang batas keyakinan minimum 50%. Hasil deteksi disimpan dalam variabel *results*, kemudian *results[0].boxes* digunakan untuk mengambil semua *bounding box* wajah yang terdeteksi. Jumlah wajah diperoleh dari len(*boxes*). Setiap *bounding box* diekstrak koordinatnya (xyxy) dan nilai *confidence*-nya, lalu digambar kotak hijau menggunakan cv2.rectangle serta ditulis label *confidence* menggunakan cv2.putText. Jumlah wajah juga ditampilkan di pojok kiri atas layar. Terakhir, *frame* yang sudah diproses ditampilkan dengan cv2.imshow, dan program akan berhenti jika tombol 'q' ditekan, kemudian melepas *webcam* dengan cap.release() dan menutup semua jendela dengan cv2.destroyAllWindows().

3.7. Diagram Alir

Untuk memudahkan pemahaman terhadap alur kerja sistem, dibuatlah sebuah diagram alir yang menggambarkan proses deteksi jumlah wajah secara berurutan. Secara garis besar, sistem memulai pekerjaan dengan mengaktifkan *webcam* laptop melalui fungsi bawaan OpenCV. Setelah kamera siap, model YOLOv8n-face.pt yang sudah disediakan sebelumnya dimuat ke dalam program.

Selanjutnya sistem memasuki tahap utama, yaitu membaca *frame* demi *frame* dari kamera. Setiap *frame* yang diperoleh langsung diproses oleh model YOLO untuk mencari area yang diduga sebagai wajah. Model hanya akan menganggap suatu area sebagai wajah jika tingkat kepercayaan mencapai 0,5 atau lebih. Apabila ada wajah yang terdeteksi, sistem akan menggambar kotak hijau di sekitar area tersebut dan menampilkan angka kepercayaan di atas kotak. Sistem juga menghitung berapa banyak wajah yang muncul dalam satu *frame*.

Setelah semua proses deteksi selesai, hasilnya ditampilkan ke layar monitor dalam bentuk video yang sudah dilengkapi kotak pembatas serta informasi jumlah wajah. Program terus berjalan secara berulang selama *webcam* masih aktif dan pengguna cukup menekan tombol 'q' pada *keyboard*. Setelah tombol tersebut ditekan, sistem akan melepaskan akses *webcam* dan menutup semua jendela tampilan.



Gambar 1. Diagram Alir Sistem Deteksi Wajah

3.8. Spesifikasi Lingkungan Pengembangan Sistem

Tabel 1. Spesifikasi Lingkungan Pengembangan Sistem

No	Komponen	Spesifikasi
1	Bahasa Pemrograman	Python 3.9
2	IDE	Spyder (dari Anaconda)
3	Library	Ultralytics (YOLOv8), OpenCV, gdown
4	Model Deteksi	YOLOv8n-face.pt
5	Sistem Operasi	Windows 10/11 64-bit
6	Prosesor	Intel Core i5 / AMD Ryzen 5 (setara)
7	RAM	8 GB
8	Webcam	Built-in laptop (resolusi 720p)

4. HASIL DAN PEMBAHASAN

4.1 Hasil Implementasi Sistem

Sistem deteksi jumlah wajah menggunakan metode YOLO telah berhasil diimplementasikan pada laptop dengan spesifikasi seperti yang tercantum pada Tabel 1. Program dibangun menggunakan bahasa pemrograman Python dengan bantuan *library* OpenCV dan model YOLOv8n-face.pt. *Webcam* bawaan laptop digunakan sebagai media input video secara *real-time*.

Saat program dijalankan, sistem melakukan inisialisasi *webcam* dan memuat model YOLOv8n-face.pt. Setelah kedua proses tersebut selesai, muncul sebuah jendela baru yang menampilkan video secara langsung dari *webcam*. Sistem berhasil mendeteksi wajah yang muncul di depan kamera dengan menandai setiap wajah menggunakan kotak pembatas (*bounding box*) berwarna hijau. Di atas setiap kotak pembatas, sistem menampilkan nilai *confidence* yang menunjukkan seberapa yakin model terhadap hasil deteksi tersebut. Selain itu, di pojok kiri atas jendela tampilan, sistem menampilkan informasi mengenai

jumlah wajah yang terdeteksi dalam setiap *frame*.

Sistem bekerja dengan cukup responsif karena proses deteksi dilakukan secara langsung pada setiap *frame* yang ditangkap oleh *webcam*. Ketika pengguna menekan tombol 'q' pada *keyboard*, program akan berhenti, melepaskan akses *webcam*, dan menutup semua jendela tampilan.



Gambar 2. Hasil Deteksi Wajah

4.2 Pengujian Sistem

Pengujian sistem dilakukan untuk mengetahui sejauh mana kinerja sistem deteksi jumlah wajah yang telah dibangun. Pengujian meliputi tiga skenario utama, yaitu variasi jumlah wajah, variasi kondisi pencahayaan, dan variasi jarak deteksi. Setiap skenario diulang sebanyak tiga kali untuk mendapatkan data yang lebih akurat.

Tabel 2.

Hasil Pengujian Berdasarkan Jumlah Wajah

Jumlah wajah	Rata-rata terdeteksi	Tingkat Akurasi	Rata-rata Confidence
1	1,0	100%	0,89
2	1,9	95%	0,87
3	2,8	93,3%	0,85
4	3,7	92,5%	0,83
5	4,5	90%	0,81

Tabel 3.

Hasil Pengujian Berdasarkan Kondisi Pencahayaan (dengan 3 wajah)

Kondisi Pencahayaan	Rata-rata terdeteksi	Tingkat Akurasi
Sangat Terang	2,9	96,7%
Normal	2,8	93,3%

Redup	2,1	70%
Gelap	0,4	13,3%

Tabel 4.

Hasil Pengujian Berdasarkan Jarak Deteksi (dengan 1 wajah)

Jarak (meter)	Rata-rata terdeteksi	Tingkat Akurasi	Rata-rata Confidence
0,5	1,0	100%	0,92
1,0	1,0	100%	0,89
1,5	1,0	100%	0,85
2,0	0,9	90%	0,78
2,5	0,7	70%	0,65
3,0	0,4	40%	0,55

4.3 Pembahasan

Berdasarkan pengujian pada tiga skenario (Tabel 2-4), sistem deteksi wajah YOLOv8n-face.pt menunjukkan kinerja yang sangat bergantung pada kondisi lingkungan. Pada variasi jumlah wajah, akurasi menurun dari 100% (1 wajah) menjadi 90% (5 wajah) akibat okklusi dan keterbatasan resolusi *webcam*, sebanding dengan YOLO-face (92,5%) dan YOLO-faceV2 (94%). Faktor pencahayaan menjadi determinan paling kritis, dengan akurasi 96,7% pada kondisi terang, turun drastis menjadi 70% saat redup, dan hanya 13,3% dalam gelap konsisten dengan YOLO5face (68% pada redup) namun lebih rendah dari YOLO-faceV2 (82%) yang menggunakan kamera eksternal berkualitas tinggi. Jarak deteksi optimal berada pada rentang 0,5-1,5 meter (akurasi 100%), menurun menjadi 40% pada jarak 3 meter karena ukuran wajah <60x60 piksel, sejalan dengan temuan YOLO-face dan Ultralytics yang menyatakan YOLOv8 mampu mendeteksi hingga 5 meter dengan kamera 4K, sehingga keterbatasan penelitian ini lebih disebabkan resolusi *webcam* 720p. Sistem memiliki kelebihan *real-time* (28-30 FPS), akurasi tinggi pada kondisi ideal (93-100%), model ringan (6 MB), dan deteksi multi-wajah baik (90-95%), namun kelemahan utamanya adalah ketergantungan ekstrem pada pencahayaan, jarak optimal terbatas ($\leq 1,5$ meter), kurang baik untuk wajah *profile*, dan masih ada *false positive*. Analisis implementasi menunjukkan *threshold confidence* 0,5 memberikan keseimbangan optimal antara deteksi dan *false positive*, dengan efisiensi

komputasi yang baik untuk laptop spesifikasi menengah. Implikasinya, sistem ini layak diimplementasikan untuk absensi otomatis dan monitoring ruangan berbiaya rendah, namun pengembangan selanjutnya direkomendasikan menggunakan kamera resolusi lebih tinggi (1080p/4K) atau pelatihan ulang model dengan data wajah dalam kondisi gelap dan berbagai pose [5].

5. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, sistem deteksi wajah menggunakan metode YOLO berhasil diimplementasikan dengan bahasa pemrograman Python, OpenCV, dan model YOLOv8n-face.pt, yang mampu berjalan secara *real-time* dengan kecepatan 28-30 *frame* per detik sehingga cukup responsif untuk pemantauan langsung. Sistem menunjukkan kinerja yang baik pada kondisi ideal, dengan deteksi satu wajah pada jarak 0,5-1,5 meter dan pencahayaan normal mencapai akurasi 100%, serta deteksi hingga lima wajah secara bersamaan masih mempertahankan akurasi di atas 90%. Namun, terdapat tiga faktor utama yang mempengaruhi kinerja sistem, yaitu pencahayaan sebagai faktor paling signifikan dengan akurasi menurun drastis hingga 13% pada kondisi redup atau gelap, jarak deteksi optimal yang terbatas pada rentang 0,5-1,5 meter dengan akurasi mulai menurun pada jarak di atas 2 meter, dan peningkatan jumlah wajah dalam satu *frame* yang cenderung menurunkan akurasi akibat adanya wajah yang saling menutupi (okklusi). Meskipun sistem telah cukup handal untuk deteksi wajah tunggal maupun beberapa wajah dalam kondisi ideal, untuk kondisi yang kurang mendukung seperti ruang gelap atau jarak jauh, sistem masih memerlukan pengembangan lebih lanjut. Berdasarkan kelemahan yang ditemukan selama penelitian, terdapat beberapa saran untuk pengembangan ke depannya, yaitu menggunakan *webcam* dengan resolusi lebih tinggi (1080p atau 4K) agar deteksi pada jarak jauh lebih akurat, menambahkan fitur pelacakan wajah (*face tracking*) agar sistem tetap dapat mengenali wajah meskipun orang bergerak atau memutar kepala, melatih ulang model YOLO dengan data wajah yang lebih bervariasi termasuk wajah dalam posisi menyamping dan kondisi cahaya redup, serta mengembangkan

sistem lebih lanjut untuk keperluan spesifik seperti absensi otomatis atau keamanan ruangan dengan menambahkan fitur pencatatan log atau database.

TERIMA KASIH

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan hidayah-Nya sehingga penelitian ini dapat diselesaikan dengan baik. Penulis mengucapkan terima kasih kepada orang tua tercinta, Nazma Bahadjai atas doa, dukungan, dan pengorbanan yang diberikan.

Terima kasih juga kepada Bapak Didi Susilo Budi Utomo dan Bapak Rihartanto selaku dosen pembimbing atas bimbingan, arahan, dan masukan selama proses penelitian, serta kepada seluruh dosen Program Studi Teknik Komputer Politeknik Samarinda atas ilmu yang telah diberikan.

Penulis juga menyampaikan apresiasi kepada teman-teman yang telah memberikan semangat dan dukungan. Semoga penelitian ini bermanfaat bagi pengembangan ilmu pengetahuan, khususnya di bidang deteksi wajah dan computer vision, serta segala kebaikan yang diberikan mendapat balasan dari Tuhan Yang Maha Esa.

DAFTAR PUSTAKA

- [1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788. <https://10.1109/CVPR.2016.91>.
- [2] J. Redmon, A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263-7271. <https://10.1109/CVPR.2017.690>.
- [3] J. Redmon, A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018. <https://10.48550/arXiv.1804.02767>.
- [4] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*. <https://10.48550/arXiv.2004.10934>.
- [5] Ultralytics, "YOLOv8: A State-of-the-Art Object Detection Model," *GitHub Repository*, 2023. [Online].

- Available: <https://github.com/ultralytics/ultralytics>.
- (JITET), vol. 12, 2024.
<https://10.23960/jitet.v13i1.5553>.
- [6] G. Jocher et al., "ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai Integrations," *Zenodo*, 2022. <https://10.5281/zenodo.7002879>.
 - [7] W. Chen, H. Huang, S. Peng, C. Zhou, and C. Zhang, "YOLO-face: a real-time face detector," *The Visual Computer*, vol. 37, no. 4, pp. 805-813, 2021. <https://doi.org/10.1007/s00371-020-01831-7>.
 - [8] Z. Yu, H. Huang, W. Chen, Y. Su, Y. Liu, and X. Wang, "YOLO-FaceV2: A scale and occlusion aware face detector," *Pattern Recognition*, vol. 155, p. 110714, Nov. 2024. <https://doi.org/10.1016/j.patcog.2024.110714>.
 - [9] D. Qi, W. Tan, Q. Yao, and J. Liu, "YOLO5Face: Why Reinventing a Face Detector," in *Computer Vision – ECCV 2022 Workshops*, Cham, 2023, pp. 228-244. https://doi.org/10.1007/978-3-031-25072-9_15.
 - [10] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A Face Detection Benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 5525-5533. <https://10.1109/CVPR.2016.596>.
 - [11] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318-327, 2020. <https://10.1109/TPAMI.2018.2858826>.
 - [12] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *arXiv preprint arXiv:1905.11946*, 2020. <https://10.48550/arXiv.1905.11946>.
 - [13] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision*, 2564–2571. <https://10.1109/ICCV.2011.6126544>.
 - [14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017. <https://doi.org/10.1109/TPAMI.2016.2577031>.
 - [15] F. H. Arby, I. Husni, and A. Amin, "Implementasi Model YOLO V5 untuk Deteksi Korek Api dalam Keamanan Penerbangan," *Jurnal Informatika dan Teknik Elektro Terapan*